

 Stagnant Standards Track: Core

EIP-210: Blockhash refactoring

Authors Vitalik Buterin (@vbuterin)**Created** 2017-02-10

Table of Contents

- Summary
- Parameters
- Specification
- BLOCKHASH_CONTRACT_CODE
- Rationale

Summary

Stores blockhashes in the state, reducing the protocol complexity and the need for client implementation complexity in order to process the BLOCKHASH opcode. Also extends the range of how far back blockhash checking can go, with the side effect of creating direct links between blocks with very distant block numbers, facilitating much more efficient initial light client syncing.

Parameters

- `CONSTANTINOPLE_FORK_BLKNUM`: TBD
- `SUPER_USER`: $2^{160} - 2$
- `BLOCKHASH_CONTRACT_ADDR`: 0xf0 (ie. 240)
- `BLOCKHASH_CONTRACT_CODE`: see below

Specification

If `block.number == CONSTANTINOPLE_FORK_BLKNUM`, then when processing the block, before processing any transactions set the code of `BLOCKHASH_CONTRACT_ADDR` to `BLOCKHASH_CONTRACT_CODE`.

If `block.number >= CONSTANTINOPLE_FORK_BLKNUM`, then when processing a block, before processing any transactions execute a call with the parameters:

- `SENDER`: `SUPER_USER`
- `GAS`: 1000000
- `TO`: `BLOCKHASH_CONTRACT_ADDR`
- `VALUE`: 0
- `DATA`: <32 bytes corresponding to the block's prevhash>

If `block.number >= CONSTANTINOPLE_FORK_BLKNUM + 256`, then the BLOCKHASH opcode instead returns the result of executing a call (NOT a transaction) with the parameters:

- `SENDER` :
- `GAS` : 1000000
- `TO` : BLOCKHASH_CONTRACT_ADDR
- `VALUE` : 0
- `DATA` : 32 byte zero-byte-leftpadded integer representing the stack argument with which the opcode was called

Also, for blocks where `block.number >= CONSTANTINOPLE_FORK_BLKNUM`, the gas cost is increased from 20 to 800 to reflect the higher costs of processing the algorithm in the contract code.

BLOCKHASH_CONTRACT_CODE

The Serpent source code is:

```
with offset = 0:
    if msg.sender == 0xfffffffffffffffffffffffffffffffffffffe:
        with bn = block.number - 1:
            while bn:
                ~sstore(offset + ~mod(bn, 256), ~calldataload(0))
                if ~mod(bn, 256):
                    ~stop()
                bn = ~div(bn, 256)
                offset += 256
            elif ~calldataload(0) >= 0 and ~calldataload(0) < block.number:
                with tbn = ~calldataload(0):
                    with dist_minus_one = block.number - tbn - 1:
                        while dist_minus_one >= 256 && ~mod(tbn, 256) == 0:
                            offset += 256
                            tbn = ~div(tbn, 256)
                            dist_minus_one = ~div(dist_minus_one, 256)
                        if dist_minus_one >= 256:
                            return(0)
                        return(~sload(offset + ~mod(tbn, 256)))
            else:
                return(0)
```

The EVM init code is:

```
0x6100f58061000e60003961010356600073fffffffffffffffffffffffffffffffffffffe33141561
```

The EVM bytecode that the contract code should be set to is:

```
0x600073ffffffffffffffffffffffffffffffffffffffffe33141561005857600143035b801561005257
```

Rationale

This removes the need for implementations to have an explicit way to look into historical block hashes, simplifying the protocol definition and removing a large component of the “implied state” (information that is technically state but is not part of the state tree) and thereby making the protocol more “pure”. Additionally, it allows blocks to directly point to blocks far behind them, which enables extremely efficient and secure light client protocols.

Citation

Please cite this document as:

Vitalik Buterin (@vbuterin), "EIP-210: Blockhash refactoring [DRAFT]," *Ethereum Improvement Proposals*, no. 210, February 2017. [Online serial]. Available: <https://eips.ethereum.org/EIPS/eip-210>.